Schema Query Containment

Melisachew Wudagae Chekol

Schema Query Containment

Melisachew Wudage Chekol

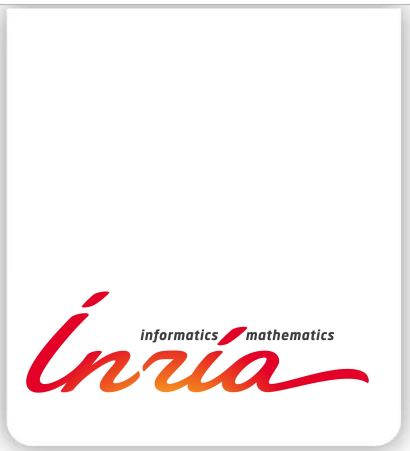
RESEARCH REPORT

N° 8484

February 2014

Project-Teams ORPAILLEUR

ISSN 0249-6399 ISRN INRIA/RR--8484--FR+ENG



Schema Query Containment

Melisachew Wudage Chekol*

Project-Teams ORPAILLEUR

Research Report n° 8484 — February 2014 — 24 pages

Abstract: SPARQL is a schema query language allowing access to the TBox part of a knowledge base. Moreover its entailment regimes enable to take into account knowledge inferred from persistently stored knowledge bases in the query answering process. Thus, the emergence of SPARQL entailment regimes provide a new perspective for the containment problem. As one has to deal with axiomatic triples, datatype reasoning, and blank nodes that result in infinite answers. Of particular interest for us is the union of conjunctive queries that are a core fragment of SPARQL. In this paper, we study the containment of such queries based on the OWL-$\mathcal{ALCH}$ Direct and RDF-Based Semantics entailment regimes.

Key-words: Query containment, schema, SPARQL, ALCH, entailment regime

* INRIA

RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Inclusion de requêtes schema

Résumé :

Mots-clés : inclusion de requêtes, SPARQL, ontologies, RDF

Contents

1 Introduction **4**

2 Preliminaries **4**
2.1 OWL-$\mathcal{ALCH}$ 5
2.2 SPARQL 5
2.3 μ-calculus 8
2.3.1 RDF Transition Systems 8

3 Schema Query Containment **9**
3.1 Containment under OWL-$\mathcal{ALCH}$ Direct Semantics Entailment Regime 10
3.1.1 Encoding OWL-$\mathcal{ALCH}$ Schema 11
3.1.2 Query Rewriting 12
3.1.3 Encoding Queries 13
3.2 Containment under OWL-$\mathcal{ALCH}$ RDF-Based Semantics Entailment Regime . . . 17
3.2.1 Encoding OWL-$\mathcal{ALCH}$ rules and schema 18

4 Discussion **20**

5 Related Work **21**

6 Conclusion **21**

1 Introduction

Most description logics query languages allow to retrieve only instance information and do not provide a means to access schema contained in a TBox. Tasks such as retrieving, accessing, and reusing parts of an ontology would be greatly facilitated by a query language that would allow querying for schema and instance information. In this regard, there exist query languages for OWL such as SPARQL, SPARQL-DL [33], SPARQL-OWL [21], and OWL-SAIQL [24]. SPARQL has support for querying schema. It allows querying of class instances, class and role names, subclasses, subproperties, domain and range relations among others. SPARQL's basic graph pattern is designed in such a way that it can be extended to query different ontology languages. SPARQL is based on simple graph matching (i.e., based on simple entailment) and is not able to include implicitly stored answers that require reasoning. Thus, in order to be able to query ontologies other than based on simple entailment. It is necessary to extend the entailment regime for the respective ontology language. This allows to generate more results that are inferred using the semantics of the axioms in the ontology. Consequently, the W3C has worked towards this issue making SPARQL a query language for ontologies also known as *SPARQL entailment regimes* [14].

Query containment is a well studied problem dating back to the end of the 70s, that started with the pioneering paper of Chandra and Merlin [6]. In that paper, it is proved that conjunctive query containment is NP-complete. In general, query containment is the problem of checking whether the result of one query is included in the result of another one for any given knowledge base. Equivalence and satisfiability problems can be derived from containment, thus, we mainly concentrate on the containment problem. In relational databases, union of conjunctive query containment has been studied using containment mappings also known as graph homomorphism and canonical databases. It is known that, for (union of) conjunctive queries, query answering and containment are equivalent problems since query containment can be reduced to query answering [6]. Unfortunately, to apply these techniques to a query language equipped with ontologies and regular expressions is not entirely possible. That is why for semistructured data query languages (referred as regular path queries) automata theoretic notions are often employed to address containment and other problems [5]. In addition to using automata, containment has been addressed by a reduction to satisfiability test. In this direction, queries are translated into formulas in a particular logic that supports the query languages features and then the overall problem is reduced into satisfiability test. Several works exist that developed and used this technique [4, 5, 7, 8, 12] which also inspired this work.

The study of SPARQL query containment has attracted a lot of attention. Notable, the works in [7] and [8] address containment under the ρdf [29] entailment regime and $\mathcal{SHI}$ schema axioms respectively and establish a double exponential upper bound complexity. Additionally, in [25] the containment and optimization of OPTIONAL queries is investigated while providing a Π_2^P-complete complexity for containment. Importantly, a benchmark of containment solvers is revealed in [10]. The implementations of containment solvers allow the extension of query and ontology languages. Consequently, in this paper, we consider a fragment of OWL 2 based on the description logic $\mathcal{ALCH}$ that can be encoded in the μ-calculus extended with nominals and converse, to study the containment of SPARQL queries under OWL-$\mathcal{ALCH}$ Direct and RDF-Based Semantics entailment regimes.

2 Preliminaries

We assume basic familiarity with the syntax and semantics of RDF(S) and OWL on the level of the RDF and OWL Primers [17, 18, 28]. We present a very minimal introductory of SPARQL,

μ-calculus, and RDF transition systems. Additionally, we shorten (in parenthesis) RDF(S) and OWL vocabularies names as follows:

`rdf:type` (`type`)	`rdfs:subClassOf` (`sc`)
`rdfs:subPropertyOf` (`sp`)	`rdfs:domain` (`dom`)
`rdfs:range` (`range`)	`rdfs:Class` (`Class`)
`owl:sameAs` (`sameAs`)	`owl:differentFrom` (`df`)
`owl:equivalentProperty` (`eqp`)	`owl:Nothing` (`Nothing`)
`owl:intersectionOf` (`inter`)	`owl:unionOf` (`union`)
`owl:someValuesFrom` (`svf`)	`owl:allValuesFrom` (`avf`)
`owl:onProperty` (`onp`)	`owl:complementOf` (`comp`)
`owl:oneOf` (`oneOf`)	`owl:hasValue` (`hasValue`)

2.1 OWL-$\mathcal{ALCH}$

Encoding OWL 2 schema axioms (built from the DL underlying OWL, $\mathcal{SROIQ}(D)$) into the μ-calculus with nominals and inverse leads to undecidability. Thus, we need to consider a fragment of OWL that alleviates this problem. Removing the features such as role inverse ($\mathcal{I}$), role composition ($\mathcal{R}$), transitivity ($\mathcal{S}$), data type restrictions (D), (qualified) cardinality restrictions ($\mathcal{Q}$) from $\mathcal{SROIQ}(D)$ reduces the complexity from N2ExpTime to ExpTime. The OWL fragment based on this logic ($\mathcal{ALCH}$) can be encoded into μ-calculus. A DL-based syntax of this language is discussed as follows. In the schema language $\mathcal{ALCH}$, a concept C can be a bottom concept ($\top$), an atomic concept A, or a complex concept $\neg C$, $\exists R.C$, $\forall R.C$ or $C \sqcap D$. A role R is an atomic role. An $\mathcal{ALCH}$ TBox consists a set of concept and role inclusion axioms [19].

Semantics: the *OWL 2 Direct Semantics (DS)* is a direct model-theoretic semantics of OWL 2 which is strongly related to the semantics of description logics. Whereas the *OWL 2 RDF-Based Semantics (RBS)* is a direct extension of the RDFS semantics. It interprets RDF triples directly without the need of mapping an RDF graph into structural OWL objects [18].

Here we present the syntax and Direct Semantics of OWL-$\mathcal{ALCH}$ concepts, roles and axioms. OWL 2 has various syntactic notations, we use the user friendly functional style syntax[1]. The OWL functional style syntax is constructed as shown in Table 1 and 2.

OWL-$\mathcal{ALCH}$ axioms: as the name implies OWL-$\mathcal{ALCH}$ axioms lack role inverse, role composition, number restrictions, and description logic datatypes. This choice allows to retain satisfiability of containment under the μ-calculus fragment chosen for this study.
Next, we present the abstract syntax of a fragment of SPARQL that is used to query OWL-$\mathcal{ALCH}$ ontologies.

2.2 SPARQL

Abstract syntax: SPARQL queries are formed from query patterns which in turn are defined inductively from *path patterns*, i.e., tuple $t \in \text{UBV} \times e \times \text{UBLV}$, with V a set of variables disjoint from UBL (URIs, Blank nodes and Literals – are used to identify values such as strings, integers and dates), and e is regular path expression (knows property hierarchies[2]. Query patterns are formed according to the following definition.

Definition 1. *A query pattern q is inductively defined as:*

$$q ::= \text{UBV} \times \text{e} \times \text{UBLV} \mid q_1 \ \textit{AND}\ q_2 \mid \{q_1\}\ \textit{UNION}\ \{q_2\}$$
$$e ::= U \mid V \mid e_1/e_2 \mid e_1 \shortmid e_2 \mid e^+ \mid e^*$$

[1] `http://www.w3.org/TR/owl-primer/`
[2] `http://www.w3.org/TR/sparql11-query/#propertypaths`

Descriptions(C)	DL Syntax	DL Semantics $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$
A	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
owl:Thing	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
owl:Bottom	$\bot$	$\bot^{\mathcal{I}} = \emptyset$
ObjectUnionOf(C_1 C_2)	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
ObjectIntersectionOf(C_1 C_2)	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \sqcap C_2^{\mathcal{I}}$
ObjectComplementOf(C)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
ObjectOneOf(o_1 o_2)	$\{o_1\} \sqcup \{o_2\}$	$\{o_1^{\mathcal{I}}\} \sqcup \{o_2^{\mathcal{I}}\}$
ObjectSomeValuesFrom(R C)	$\exists R.C$	$\{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$
ObjectAllValuesFrom(R C)	$\forall R.C$	$\{x \mid \forall y.\langle x, y\rangle \in R^{\mathcal{I}}$ implies $y \in C^{\mathcal{I}}\}$
Descriptions(R)		
R	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Individuals (o)		
o	o	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
_:*anonymous*	_:*anonymous*	_:$anonymous^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

Table 1: OWL-$\mathcal{ALCH}$ syntax: concept, role and individual constructs.

OWL syntax	DL syntax	DL Semantics $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \sqsubseteq C_2^{\mathcal{I}}$
EquivalentClasses(C_1 C_2)	$C_1 \equiv C_2$	$C_1^{\mathcal{I}} \equiv C_2^{\mathcal{I}}$
DisjointClasses(C_1 C_2)	$C_1 \sqcap C_2 \sqsubseteq \bot$	
SubObjectPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \sqsubseteq R_2^{\mathcal{I}}$
EquivalentObjectProperties(R_1 R_2)	$R_1 \equiv R_2$	$R_1^{\mathcal{I}} \equiv R_2^{\mathcal{I}}$
ObjectPropertyDomain(R C)	$\exists R.\top \sqsubseteq C$	
ObjectPropertyRange(R C)	$\top \sqsubseteq \forall R.C$	
ClassAssertion(C o)	$o : C$	$o^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
ObjectPropertyAssertion(R o_1 o_2)	$(o_1, o_n) : R$	$(o_1^{\mathcal{I}}, o_n^{\mathcal{I}}) \subseteq R^{\mathcal{I}}$

Table 2: OWL-$\mathcal{ALCH}$ axioms syntax and semantics.

The AND fragment of query patterns are also known as *basic graph patterns (BGPs)* . A SPARQL SELECT query is a query of the form $q(W)$ such that W is a tuple of variables in V which are called *distinguished variables*, and q is a query pattern. The variables V $\setminus$ W are called *non-distinguished variables*. As discussed below, in SPARQL entailment regimes, non-distinguished variables cannot appear by design.

Semantics: SPARQL has multiset (or bag) semantics, however, when dealing with containment, we consider set semantics. This is due to the undecidability of union of conjunctive queries under bag semantics [20]. We refer the reader for the semantics SPARQL queries under OWL DS and RBS entailment regimes to [14]. Importantly, under both DS and RBS entailment regimes, there are no non-distinguished variables since answers are computed from the BGPs of the query, projection is considered a post-processing step. In other words, projection is performed after the evaluation of basic graph patterns, i.e., after all the variables are bound to values in the ontology. We consider this essential feature when dealing with containment.

Now, we provide a compact summary of the semantics of SPARQL query evaluation under OWL-$\mathcal{ALCH}$ Direct Semantics entailment regime. A detailed discussion with insightful examples can be found in [14, 16, 21]. The following definition is taken from [14].

Definition 2 (SPARQL query evaluation under OWL DS entailment regime)**.** *A partial mapping function ρ is a solution for a Object Graph Pattern (OGP) P and G under OWL Direct Semantics entailment if:*

1. *G can be mapped into an OWL DL ontology $O(G)$*
2. *P can be mapped into an extended OWL DL axioms $O(P)$*
3. *Domain of ρ is exactly the set of variables in P*
4. *Terms in the range of ρ occur in $O(G)$ or Voc(OWL): OWL vocabulary terms*
5. *If $O(P')$ obtained from $O(P)$ by replacing anonymous individuals (blank nodes) with either IRIs or blank nodes is such that:*
   $O(G) \cup O(P')$ is an OWL DL ontology and $sk(O(G)) \models_{DS} sk(\rho(O(P')))$

The function $sk(.)$ replaces blank nodes with fresh IRIs (IRIs that are neither in the queried graph nor in the query). $\models_{DS}$ denotes the OWL DS entailment relation.

Definition 3 (Containment under an entailment regime)**.** *Given a set of TBox axioms $\mathcal{S}$ of OWL-$\mathcal{ALCH}$ and two queries q and q' with the same arity, q is contained in q' with respect to an entailment regime e, denoted $q \sqsubseteq_e q'$, iff $q(\mathcal{O}) \subseteq q'(\mathcal{O})$ for every ontology $\mathcal{O}$ satisfying $\mathcal{S}$ based on e-semantics. Where $q(\mathcal{O})$ denotes the answers of q when evaluated in $\mathcal{O}$.*

Definition 4 (Equivalence)**.** *Given queries q, q', and a set of axioms $\mathcal{S}$, q and q' are equivalent with respect to an entailment regime e, denoted $q \equiv_e q'$, iff $q \sqsubseteq_e q'$ and $q' \sqsubseteq_e q$.*

Example 1 (Containment under e-entailment)**.** *Containment does not hold between the following queries, namely q and q', under simple and RDF entailment. However, containment holds under e- (RDFS, DS, and RBS) entailment regime in one direction, i.e., $q \sqsubseteq_e q'$ and $q' \not\sqsubseteq_e q$.*

```
SELECT ?s ?c WHERE {
  ?s  ?x  ?d .
  ?x  sp  type .
  ?d  sc  ?c .
}
```

```
SELECT ?s ?c
 WHERE {
  ?s type ?c .
}
```

Example 2. *Consider containment of the following queries under different entailment regimes.*

$$\begin{aligned} q(x,y) &= (z, \texttt{sp}, \texttt{sc}) \textsc{ and } (z, \texttt{domain}, \texttt{Class}) \textsc{ and } (x, z, y) \\ q'(x,y) &= (x, \texttt{sc}, y) \\ q''(x,y) &= (x, \texttt{type}, \texttt{Class}) \textsc{ union } (y, \texttt{type}, \texttt{Class}) \end{aligned}$$

Example 3 (Containment)**.** *q is contained in q' under the RDFS entailment regime but not under the simple and RDF regimes.*

$$\begin{aligned} q(x) &= (x, sp, \text{parentOf}) \\ q'(x) &= (x, a, \text{rdf} : \text{Property}) \end{aligned}$$

Note that we switch between SPARQL's W3C and abstract syntax of queries as necessary.

The evaluation of SPARQL queries is proved to be PSPACE-complete. However, the evaluation problem is NP-complete for the fragment containing only AND and UNION query patterns [1]. In addition, union of conjunctive query answering in $\mathcal{ALCH}$ is ExpTime [31]. SPARQL queries can be encoded into μ-calculus formula. The syntax of this logic is given in the next section.

2.3 μ-calculus

The μ-calculus is a logic obtained by adding fixpoint operators to ordinary modal logic [22]. It has several extensions, for this work, we use the μ-calculus with nominals and converse programs. The syntax of the μ-calculus is composed of countable sets of *atomic propositions* and *nominals* AP, a set of *variables* Var, a set of *programs and their respective converses* Prog for navigating in graphs. A μ-calculus formula, φ, can be defined inductively as follows:

$$\varphi \quad ::= \quad \top \mid q \mid X \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \mu X \varphi \mid \nu X \varphi$$

where $q \in AP, X \in Var$ and $a \in Prog$ is a transition program or its converse $\bar{a}$. The greatest and least fixpoint operators (ν and μ) respectively introduce general and finite recursion in graphs [22]. The semantics of the μ-calculus is given over a transition system, $K = (S, R, L)$ where S is a non-empty set of nodes, $R : Prog \rightarrow 2^{S \times S}$ is the transition function, and $L : AP \rightarrow 2^S$ assigns a set of nodes to each atomic proposition or nominal where it holds, such that $L(p)$ is a *singleton* for each nominal p. For converse programs, R can be extended as $R(\bar{a}) = \{(s', s) \mid (s, s') \in R(a)\}$. The valuation function $V : \text{Var} \rightarrow 2^S$ maps each variable into a set of nodes. For a valuation V, variable X, and a set of nodes $S' \subseteq S$, $V[X/S']$ is the valuation that is obtained from V by assigning S' to X. The semantics of a formula in terms of a transition system K (a.k.a. Kripke structure) and a valuation function is represented by $[\![\varphi]\!]^K_V \subseteq S$. The semantics of basic μ-calculus formulae is defined as follows:

$$\begin{aligned}
[\![\top]\!]^K_V &= S \qquad [\![\bot]\!]^K_V = \emptyset \\
[\![q]\!]^K_V &= L(q), q \in AP,\ L(q) \text{ is singleton if } q \text{ is a nominal} \\
[\![X]\!]^K_V &= V(X), X \in Var \qquad [\![\neg\varphi]\!]^K_V = S \backslash [\![\varphi]\!]^K_V \\
[\![\varphi \wedge \psi]\!]^K_V &= [\![\varphi]\!]^K_V \cap [\![\psi]\!]^K_V \\
[\![\varphi \vee \psi]\!]^K_V &= [\![\varphi]\!]^K_V \cup [\![\psi]\!]^K_V \\
[\![\langle a \rangle \varphi]\!]^K_V &= \{s \in S \mid \exists s' \in S.(s, s') \in R(a)\ \wedge s' \in [\![\varphi]\!]^K_V\} \\
[\![[a]\varphi]\!]^K_V &= \{s \in S \mid \forall s' \in S.(s, s') \in R(a) \Rightarrow s' \in [\![\varphi]\!]^K_V\} \\
[\![\mu X \varphi]\!]^K_V &= \bigcap \{S' \subseteq S \mid [\![\varphi]\!]^K_{V[X/S']} \subseteq S'\} \\
[\![\nu X \varphi]\!]^K_V &= \bigcup \{S' \subseteq S \mid S' \subseteq [\![\varphi]\!]^K_{V[X/S']}\}
\end{aligned}$$

Note that the evaluation of sentences is independent of valuations and hence we define the following. For a sentence φ, a Kripke structure $K = (S, R, L)$, and $s \in S$, we denote $K, s \models \varphi$ if and only if $s \in [\![\varphi]\!]^K$, henceforth K is considered as a *model* of φ. In other words, K is considered as a model of ϕ if there exists an $s \in S$ such that $K, s \models \phi$. If a sentence has a model, then it is called *satisfiable*. If a μ-calculus formula ψ appears under the scope of a least μ or greatest ν fixed point operator over the programs $\{s, p, o, d, \bar{s}, \bar{p}, \bar{o}, \bar{d}\}$ as, $\mu X.\psi \vee \langle s \rangle X \vee \langle p \rangle X \vee \cdots$ or $\nu X.\psi \wedge \langle s \rangle X \wedge \langle p \rangle X \wedge \cdots$, then, for the sake of legibility, we denote the formulae by $lfp(X, \psi)$ and $gfp(X, \psi)$, respectively. μ-calculus formulas that are encodings of SPARQL queries are interpreted over RDF transition systems.

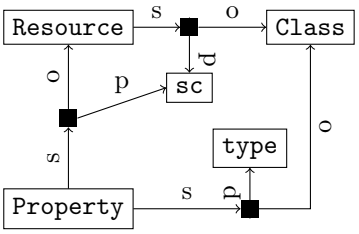
2.3.1 RDF Transition Systems

RDF transition systems are first introduced in [7].They are labelled transition system representations of RDF graphs where two sets of nodes are introduced: one set for each triple (called triple node) and the other set for each subject, predicate, and object of each triple. A triple node

is connected to its subject, predicate, and object nodes. Transition from one node to another is done by using a set of transition programs $\{s, p, o, d\}$ and their converses.

Example 4 (RDF transition system)**.** *Consider the RDF transition system associated with RDFS graph given below.*

```
Resource sc Class .
Resource sc Property .
Property type Class .
```



The μ-calculus extended with nominals and converse lacks functionality or number restrictions. Thus, one cannot impose that each triple node is connected to exactly one node for each subject, predicate, and object node. However, one can impose a lighter restriction to achieve this by taking advantage of the technique introduced in [13] and adopted in [9]. Since it is not possible to ensure that there is only one successor, then we restrict all the successors to bear the same constraints. Thus, they become interchangeable. This can be done by rewriting the formulas using a function f such that all occurrences of $\langle a\rangle\varphi$ (existential formulas) are replaced by $\langle a\rangle\top \wedge [a]\varphi$. f is defined inductively on the structure of a μ-calculus formula.

$$
\begin{aligned}
f(\top) &= \top \\
f(q) &= q \quad q \in AP \cup Nom \\
f(X) &= X \quad X \in Var \\
f(\neg\varphi) &= \neg f(\varphi) \\
f(\varphi \wedge \psi) &= f(\varphi) \wedge f(\psi) \\
f(\varphi \vee \psi) &= f(\varphi) \vee f(\psi)
\end{aligned}
\qquad
\begin{aligned}
f(\langle a\rangle\varphi) &= \langle a\rangle\top \wedge [a]f(\varphi) \quad a \in \{\bar{s}, p, o\} \\
f(\langle a\rangle\varphi) &= \langle a\rangle f(\varphi) \quad a \in \{d, s, \bar{p}, \bar{o}\} \\
f([a]\varphi) &= [a]f(\varphi) \quad a \in \text{Prog} \\
f(\mu X.\varphi) &= \mu X.f(\varphi) \\
f(\nu X.\varphi) &= \nu X.f(\varphi)
\end{aligned}
$$

Thus, when checking for query containment, we assume that the formulas are rewritten using function f.

So far, we have laid the foundations by introducing the basics of SPARQL, μ-calculus and RDF transition systems. We are ready to final embark on the problem of schema query containment.

3 Schema Query Containment

To show the importance of SPARQL query containment under entailment regimes, we provide the following examples.

Example 5. *Containment between q and q' does not hold under the RDFS entailment. However, containment holds $q \sqsubseteq_e q'$ (for e = DS and RBS entailment regimes) because* `eqp` *can be expressed as a two-way* `sp` *relation.*

$$
\begin{aligned}
q(y) &= (x, \texttt{eqp}, \texttt{sc}) \ \textit{AND}\ (y, x, C) \\
q'(y) &= \{(x, \texttt{sp}, \texttt{sc}) \ \textit{UNION}\ (\texttt{sc}, \texttt{sp}, x)\} \ \textit{AND}\ (y, x, C)
\end{aligned}
$$

Example 6 (Satisfiability)**.** *q is unsatisfiable because of the semantics of* `disjointWith`*.* $q(x) = (x, \texttt{type}, y)$ AND $(x, \texttt{type}, D)$ AND $(y, \texttt{disjointWith}, D)$

These examples show that when dealing with query containment and satisfiability problems, one has to consider not only the schema axioms but also the semantics of the schema vocabulary.

Beforehand, there are two main issues that one needs to take care of when dealing with schema query containment: infinite answers and blank nodes.

Infinite answers: query evaluation under OWL entailment regimes can result in infinite answers due to either: (1) inconsistency in the queried graph, (2) an infinite number of axiomatic triples, (3) arbitrary blank nodes, or (4) infinite entailments in datatype reasoning. The axiomatic triples of RDFS are used only by the RBS and can be handled by suitably restricting solutions to an answer domain. In fact, the W3C recommendation[3] prohibits infinite answers by restricting answers to RDF(S) and OWL vocabulary subtracting rdf:_i (for all $i \in \mathbb{N}$) and terms occurring in the data graph. Besides, blank nodes are skolemized, and inconsistencies are treated as errors [14].

Blank nodes (bnodes): in the OWL entailment regimes, there are two sets of bnodes: (i) bnodes that represent anonymous OWL individuals, and (ii) bnodes used for encoding complex OWL syntax in RDF. The DS treats these bnodes differently (in the query answering process) whereas RBS handles bnodes just like RDFS, even in cases where they are needed for encoding OWL class expressions. However, when dealing with containment, we do not distinguish between these sets of bnodes in both DS and RBS entailment regimes. They are interpreted as existential variables (or non-distinguished variables). Since non-distinguished variables are not a feature of entailment regimes, we treat them as distinguished for checking containment.

3.1 Containment under OWL-$\mathcal{ALCH}$ Direct Semantics Entailment Regime

The OWL DS maps BGPs into OWL structural objects for query answering. However, when dealing with containment, we do not perform this mapping (as it is not relevant for testing containment). Checking the containment of SPARQL queries under the DS entailment can be reduced to encoding OWL-$\mathcal{ALCH}$ ontology axioms and queries as μ-calculus formulas and checking the unsatisfiability of the encoding (or formula). Consequently, in the following, we present the encoding of OWL-$\mathcal{ALCH}$ schema axioms, rewriting queries by considering the semantics of the schema, encoding the rewritten queries, and finally the reduction of containment into unsatisfiability test.

Definition 5. *Given a set of schema axioms $\mathcal{S}$ and SPARQL queries q and q', their containment under the OWL-$\mathcal{ALCH}$ Direct semantics entailment, $q \sqsubseteq_{\text{DS}} q'$, can be reduced to μ-calculus by rewriting the queries and checking the validity of the formula $\eta(S) \wedge \mathcal{A}(\tau(q)) \wedge \neg\mathcal{A}(\tau(q'))$. η, $\mathcal{A}$, and τ are schema encoding, query encoding, and query rewriting functions respectively.*

In the following, we present each of these functions in detail.

[3] `http://www.w3.org/TR/sparql11-entailment/`

3.1.1 Encoding OWL-$\mathcal{ALCH}$ Schema

Given a set of schema axioms $\mathcal{S} = \{s_1, \ldots, s_n\}$ comprising concept and role inclusions. The μ-calculus encoding of $\mathcal{S}$ is obtained using the function η as follows:

$$\begin{aligned}
\eta(\mathcal{S}) &= \eta(s_1) \wedge \cdots \wedge \eta(s_n) \\
\eta(C_1 \sqsubseteq C_2) &= \mathit{gfp}\big(X, \eta(C_1) \Rightarrow \eta(C_2)\big) \\
\eta(\bot) &= \bot \quad \eta(A) = A \quad \eta(\neg C) = \neg\omega(C) \\
\eta(C_1 \sqcap C_2) &= \eta(C_1) \wedge \eta(C_2) \\
\eta(\exists R.C) &= \langle s\rangle\big(\langle p\rangle R \wedge \langle o\rangle(\langle s\rangle\langle o\rangle\eta(C))\big) \\
\eta(\forall R.C) &= [s]\big([p]R \Rightarrow [o]([s][o]\eta(C))\big) \\
\eta(R_1 \sqsubseteq R_2) &= \mathit{gfp}\big(X, R_1 \Rightarrow R_2\big)
\end{aligned}$$

Lemma 1. *Given a set of $\mathcal{ALCH}$ schema axioms $\mathcal{S}$, $\mathcal{S}$ has a model iff $\eta(\mathcal{S})$ is satisfiable.*

Proof. ($\Rightarrow$) assume that there exists a model $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ of $\mathcal{S}$ such that $\mathcal{I} \models \mathcal{S}$. We build a restricted transition system $K = (S, R, L)$ from $\mathcal{I}$ using the following:

- for each element of the domain $e^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, we create a node $n^e \in S'$,
- for each atomic concept A, if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, then $(n^a, t) \in R(s), (t, n^{type}) \in R(p)$, $(t, n^A) \in R(o)$, $L(type), = n^{type}, L(A) = n^A$ and $L(a) = n^a$ where $t \in S''$,
- for each atomic role T, if $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in T^{\mathcal{I}}$, then $(n^x, t) \in R(s)$, $(t, n^T) \in R(p)$, and $(t, n^y) \in R(o)$ such that $n^x, n^y, n^T \in S'$, $t \in S''$, and $L(x) = n^x$, $L(T) = n^T$, $L(y) = n^y$,
- $S = S' \cup S''$

To show that $\eta(\mathcal{S})$ is satisfiable in K. We proceed inductively on the construction of the formula. As the axioms, $\{s_1, \ldots, s_n\}$, are made of role or concept inclusions or nominals, we consider the following cases:

- when $\eta(s_i) = \mathit{gfp}\big(X, \omega(C_1) \Rightarrow \omega(C_2)\big)$.

  From the assumption it is the case that $\mathcal{I} \models c_i$, alternatively, $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. To show that $\eta(s_i)$ is satisfiable in K, we proceed on the construction of C_1 and C_2.

  1. If C_1 and C_2 are atomic concepts, then their encodings are atomic propositions C_1 and C_2 in the μ-calculus. From $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, we have that $[\![\omega(C_1)]\!]^K \subseteq [\![\omega(C_2)]\!]^K = [\![C_1]\!]^K \subseteq [\![C_2]\!]^K$. And hence, $\omega(C_1) \Rightarrow \omega(C_2)$ is satisfiable in K. Besides, the general recursion ν guarantees that the constraint is satisfied in each state of the transition system. Therefore, $\eta(s_i)$ is satisfiable.
  2. If C_1 and C_2 are complex concepts, then $K \models \eta(s_i)$ can be proved by exploiting the construction each axiom and ω. For instance, if the axiom is $\mathcal{I} \models C \wedge D \wedge \exists R.C \sqsubseteq \forall R.D$

     $\Leftrightarrow (C \wedge D \wedge \exists R.C)^{\mathcal{I}} \subseteq (\forall R.D)^{\mathcal{I}}$

     $\Leftrightarrow C^{\mathcal{I}} \cap D^{\mathcal{I}} \cap (\exists R.C)^{\mathcal{I}} \subseteq (\forall R.D)^{\mathcal{I}}$

     $\Leftrightarrow [\![\omega(C)]\!]^K \cap [\![\omega(D)]\!]^K \cap [\![\omega(\exists R.C)]\!]^K \subseteq [\![\omega(\forall R.D)]\!]^K$ from case (1).

     $\Leftrightarrow [\![\omega(C) \wedge \omega(D) \wedge \omega(\exists R.C)]\!]^K \subseteq [\![\omega(\forall R.D)]\!]^K$ μ-calculus semantics.

     $\Leftrightarrow [\![\mathit{gfp}\big(X, \omega(C) \wedge \omega(D) \wedge \omega(\exists R.C) \Rightarrow \omega(\forall R.D)\big)]\!]^K$ the gfp (or ν) ensures that the implication holds in the entire transition system.

     Intuitively, this extends to any axiom composed of complex concept constructs.

- when $\eta(s_i) = gfp\big(X, \omega(r_1) \Rightarrow \omega(r_2)\big)$. From $r_1^{\mathcal{I}} \subseteq r_2^{\mathcal{I}}$ we have that $\exists n^{r_1} \in L(r_1)$ implies $\exists n^{r_2} \in L(r_2)$ in K. Thus, $\exists s \in [\![\omega(r_1) \Rightarrow \omega(r_2)]\!]^K$. As K is a construction of $\mathcal{I}$, $\eta(c_i)$ is satisfiable in K.

Since K is a model of each $\eta(s_i)$, then $\eta(\mathcal{S})$ is satisfiable.
($\Leftarrow$) consider a transition system model K for $\eta(\mathcal{S})$. From K, we construct an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ and show that it is a model of $\mathcal{S}$.

- $\Delta^{\mathcal{I}} = S$, $A^{\mathcal{I}} = [\![A]\!]^K$ for each atomic concept A,
- $\top^{\mathcal{I}} = [\![\top]\!]^K$, for a top concept,
- $r^{\mathcal{I}} = \{(s, s') \mid \forall t \in [\![r]\!]^K \wedge t' \in S \wedge (s, t') \in R(s) \wedge (t', t) \in R(p) \wedge (t', s') \in R(o)\}$ for each atomic role r,

Consequently, formulas such as $gfp\big(X, \omega(r_1) \Rightarrow \omega(r_2)\big)$ and $gfp\big(X, \omega(C_1) \Rightarrow \omega(C_2)\big)$ are true in $\mathcal{I}$. The first formula expresses that there is no node in the transition system where $\omega(r_1)$ holds and $\omega(r_2)$ does not hold. This is equivalent to $\omega(r_1) \Rightarrow \omega(r_2)$ and $[\![r_1]\!]^K \subseteq [\![r_2]\!]^K$ since r_1 and r_2 are basic roles. Thus, we obtain $r_1^{\mathcal{I}} \subseteq r_2^{\mathcal{I}}$ and $\mathcal{I} \models r_1 \sqsubseteq r_2$.

On the other hand, for the latter formula from above, one can exploit its construction. Note however that, similar justifications as above can be worked out to arrive at $\mathcal{I} \models C_1 \sqsubseteq C_2$ if C_1 and C_2 are basic concepts. Nonetheless, if they are complex concepts, we proceed as below. Consider the case when $C_1 = A \sqcap B$ and $C_2 = \exists R.C$, $[\![\omega(C_1) \Rightarrow \omega(C_2)]\!]^K$

$$
\begin{aligned}
&\Leftrightarrow [\![\omega(A \sqcap B)]\!]^K \subseteq [\![\omega(\exists R.C)]\!]^K \\
&\Leftrightarrow [\![A \wedge B]\!]^K \subseteq [\![\langle s\rangle\big(\langle p\rangle R \wedge \langle o\rangle(\langle s\rangle\langle o\rangle C)\big)]\!]^K \\
&\Leftrightarrow [\![A]\!]^K \wedge [\![B]\!]^K \subseteq \{s \mid \exists s'.s \in [\![\langle s\rangle\langle p\rangle R]\!]^K \wedge s' \in [\![\langle s\rangle\langle o\rangle C]\!]^K\} \\
&\Leftrightarrow A^{\mathcal{I}} \cap B^{\mathcal{I}} \subseteq \{s \mid \exists s'.(s, s') \in R^{\mathcal{I}} \wedge s' \in C^{\mathcal{I}}\} \\
&\Leftrightarrow (A \sqcap B)^{\mathcal{I}} \subseteq (\exists R.C)^{\mathcal{I}} \\
&\Leftrightarrow \mathcal{I} \models C_1 \sqsubseteq C_2
\end{aligned}
$$

Accordingly, from $\mathcal{I} \models s_1 \wedge \cdots \wedge \mathcal{I} \models s_n$, it follows that $\mathcal{I} \models \mathcal{S}$. □

3.1.2 Query Rewriting

SPARQL query containment under OWL-$\mathcal{ALCH}$ DS entailment regime can be determined by rewriting queries using the semantics of the schema vocabulary and then reducing the encoding of the rewriting to unsatisfiability test. The rewriting of SPARQL queries can be done by using SPARQL property paths and the semantics of the schema vocabulary as discussed in the following.

Definition 6. *Given a SPARQL query q, a rewriting function τ produces its rewriting shown in Table 3.*

Example 7. *Consider the rewritings of the queries in Example 1.*

$$
\begin{aligned}
\tau((s,\mathtt{sp},o)) &= (s,\mathtt{sp}^+ \mid \mathtt{eqp}^+,o)\\
\tau((s,p,o)) &= (s,x,o) \text{ AND } (x,\mathtt{sp}^* \cdot \mathtt{eqp}^*,p)\\
\tau((s,\mathtt{type},o)) &= (s,\mathtt{type}/(\mathtt{sc}^* \mid \mathtt{eqc}^*),o) \text{ UNION } (s,x,y) \text{ AND}\\
&\quad (x,(\mathtt{sp}^* \mid \mathtt{eqp}^*)/\mathtt{dom}/(\mathtt{sc}^* \mid \mathtt{eqc}^*),o) \text{ UNION } (y,x,s) \text{ AND}\\
&\quad (x,(\mathtt{sp}^* \mid \mathtt{eqp}^*)/\mathtt{range}/(\mathtt{sc}^* \mid \mathtt{eqc}^*),o)\\
\tau((p,\mathtt{dom},c)) &= (p,(\mathtt{sp}^* \mid \mathtt{eqp}^*)/\mathtt{dom}/(\mathtt{sc}^* \mid \mathtt{eqc}^*),c)\\
\tau((p,\mathtt{range},c)) &= (p,(\mathtt{sp}^* \mid \mathtt{eqp}^*)/\mathtt{range}/(\mathtt{sc}^* \mid \mathtt{eqc}^*),c)\\
\tau((s,x,o)) &= (s,x,o) \text{ when } x \text{ is a variable}\\
\tau(c_1,\mathtt{disWith},c_2) &= (c_1,\mathtt{sc},\neg c_2)\\
\tau(s,\mathtt{eqc},o) &= (s,\mathtt{eqc},o) \text{ UNION } (s,\mathtt{sc},o) \text{ AND } (o,\mathtt{sc},s)\\
\tau((p,\mathtt{eqp},q)) &= (p,\mathtt{eqp}^+,q) \text{ UNION}\\
&\quad (p,\mathtt{sp},q) \text{ AND } (q,\mathtt{sp},p)\\
\tau(c_1,\mathtt{sc},c_2) &= (c_1,\mathtt{sc}^+ \mid \mathtt{eqc}^+,c_2) \text{ UNION}\\
&\quad (c_1,\mathtt{svf},y_1) \,.\, (c_1,\mathtt{onp},p) \,.\, (c_2,\mathtt{svf},y_2) \,.\\
&\quad (c_2,\mathtt{onp},p) \,.\, (y_1,\mathtt{sc},y_2) \text{ UNION}\\
&\quad (c_1,\mathtt{svf},y) \,.\, (c_1,\mathtt{onp},p_1) \,.\, (c_2,\mathtt{svf},y) \,.\\
&\quad (c_2,\mathtt{onp},p_2) \,.\, (p_1,\mathtt{sp},p_2) \text{ UNION}\\
&\quad (c_1,\mathtt{avf},y_1) \,.\, (c_1,\mathtt{onp},p) \,.\, (c_2,\mathtt{avf},y_2) \,.\\
&\quad (c_2,\mathtt{onp},p) \,.\, (y_1,\mathtt{sc},y_2) \text{ UNION}\\
&\quad (c_1,\mathtt{avf},y) \,.\, (c_1,\mathtt{onp},p_1) \,.\, (c_2,\mathtt{avf},y) \,.\\
&\quad (c_2,\mathtt{onp},p_2) \,.\, (p_1,\mathtt{sp},p_2)\\
\tau(q_1 \text{ AND } q_2) &= \tau(q_1) \text{ AND } \tau(q_2)\\
\tau(q_1 \text{ UNION } q_2) &= \tau(q_1) \text{ UNION } \tau(q_2)
\end{aligned}
$$

Table 3: Rewriting SPARQL queries using inference rules.

```
SELECT ?s WHERE
{
  ?s ?x ?d . ?x sp+ type . ?d sc+ ?c .
}
SELECT ?s WHERE
{
  { ?s type/(sc*|eqp*) ?c .} UNION
  { ?s ?r ?y . ?r sp*/dom/sc* ?c .} UNION
  { ?y ?r ?s . ?r sp*/range/sc* ?c .}
}
```

3.1.3 Encoding Queries

The variables and constants in a query are encoded using nominals of the μ-calculus. Basically, the variables and constants are frozen, i.e., equivalent to obtaining a canonical instance of the query. Afterwards, a recursive function A is used to inductively construct a formula. Given two

SPARQL queries $q(W)$ and $q'(W)$, the containment test $q(W) \sqsubseteq q'(W)$ can be encoded into the μ-calculus as follows:

$$\begin{aligned}\Phi_{q \sqsubseteq q'} &= \mathcal{A}(q) \wedge \neg \mathcal{A}(q') \\ \mathcal{A}((x,y,z)) &= \mathit{lfp}\big(X, \langle \bar{s} \rangle x \wedge \mathcal{R}(y,z)\big) \\ \mathcal{A}(q_1 \text{ AND } q_2) &= \mathcal{A}(q_1) \wedge \mathcal{A}(q_2) \\ \mathcal{A}(q_1 \text{ UNION } q_2) &= \mathcal{A}(q_1) \vee \mathcal{A}(q_2)\end{aligned}$$

In *lfp*, x and z are nominals whereas regular expression patterns (property hierarchies) that appear in the query are encoded into atomic propositions using the function $\mathcal{R}$. This function takes two arguments (the predicate which is a regular expression pattern and the object of a triple).

$$\begin{aligned}\mathcal{R}(uri, y) &= \langle p \rangle uri \ \wedge \langle o \rangle y \\ \mathcal{R}(x, y) &= \langle p \rangle x \wedge \langle o \rangle y \\ \mathcal{R}(e \mid e', y) &= (\mathcal{R}(e, y) \vee \mathcal{R}(e', y)) \\ \mathcal{R}(e/e', y) &= \mathcal{R}(e, \langle s \rangle \mathcal{R}(e', y)) \\ \mathcal{R}(e^+, y) &= \mu X. \mathcal{R}(e, y) \vee \mathcal{R}(e, \langle s \rangle X) \\ \mathcal{R}(e^*, y) &= \mathcal{R}(e^+, y) \vee \langle \bar{s} \rangle y\end{aligned}$$

Example 8 (Query encoding)**.** *Consider the encoding of the containment between queries in Example 1.*

$$\begin{aligned}\Phi_{q \sqsubseteq q'} &= \mathcal{A}(q) \wedge \neg \mathcal{A}(q') \\ \mathcal{A}(q) &= \mathcal{A}(s, x, d) \wedge \mathcal{A}(x, \mathtt{sp}, \mathtt{type}) \wedge (d, \mathtt{sc}, c) \\ &= \mathit{lfp}\big(X, \langle \bar{s} \rangle s \wedge \langle p \rangle x \wedge \langle o \rangle d\big) \wedge \mathit{lfp}\big(X, \langle \bar{s} \rangle x \wedge \langle p \rangle \mathtt{sp} \wedge \langle o \rangle \mathtt{type}\big) \wedge \\ &\qquad \mathit{lfp}\big(X, \langle \bar{s} \rangle d \wedge \langle p \rangle \mathtt{sc} \wedge \langle o \rangle c\big) \\ \mathcal{A}(q') &= \mathit{lfp}\big(X, \langle \bar{s} \rangle s \wedge \langle p \rangle \mathtt{type} \wedge \langle o \rangle c\big) \\ \neg \mathcal{A}(q') &= \mathit{gfp}\big(X, [\bar{s}] \neg s \vee [p] \neg \mathtt{type} \vee [o] \neg c\big)\end{aligned}$$

So far, we have produced the encoding of schema axioms and the rewriting of queries and their encodings. Thus, we are ready to prove the correctness of this reduction. We denote the reduction by $\Phi(q \sqsubseteq_{\mathrm{DS}} q') = \eta(\mathcal{S}) \wedge \mathcal{A}(\tau(q)) \wedge \neg \mathcal{A}(\tau(q'))$.

Lemma 2. *Let q be a SPARQL query, for every RDF transition system K whose associated RDF graph is G, we have that q is satisfiable in G iff $\mathcal{A}(q)$ is satisfiable in K.*

Proof. (Sketch) ($\Rightarrow$) $[\![q]\!]_G \neq \emptyset$ implies that G is at least a canonical instance of q and can be produced using a function f as shown below:

- if $(x, y, z) \in q$, then $f((x, y, z)) = (x, y, z) \in G$,
- if $(x, e, z) \in q$, then $f((x, e, z)) = (x, e, z) \in G$,
- if $(x, e/e', z) \in q$, then $f((x, e, y)) \in G$ and $f((y, e', z)) \in G$,
- if $(x, e \mid e', z) \in q$, then $f((x, e, z)) \in G$ or $f((x, e', z)) \in G$,
- if $(x, e^+, z) \in q$, then $f((x, e, y_1)) \in G$ and $\ldots$ and $f((y_n, e, z)) \in G$,

- if $(x, e^*, z) \in q$, then $f((x, e^+, z)) \in G$

Consequently, since G is an instance of q, G is a model of q. Now, one can construct an RDF transition system $\sigma(G) = (S, R, L)$ in the same way as is done in [9]. To prove that $\sigma(G)$ is a model of $\mathcal{A}(q)$, we consider the encoding of the non-distinguished variables with $\top$ suffices to justify that $\sigma(G)$ is a model of its encoding. Since $\top$ gets instantiated (in all possible ways) with the constants (and frozen variables) appearing in the lhs query.
($\Leftarrow$) Assume that $[\![\mathcal{A}(q)]\!]^K \neq \emptyset$. We now create an RDF graph G from K as follows:

- if $\forall s_1, s_2, s_3 \in S' \wedge t \in S''.(s_1, t) \in R(s) \wedge (t, s_2) \in R(p) \wedge (t, s_3) \in R(o)$ and for each triple $t_i = (x_i, y_i, z_i) \in q$ if $s_1 \in L(x_i) \wedge s_2 \in L(y_i) \wedge s_3 \in L(z_i)$, then $(x_i, y_i, z_i) \in G$. This case holds if x_i, y_i and z_i are either distinguished variables or constants. Note here that if x_i or y_i or z_i appear in another triple $t_j = (x_j, y_j, z_j) \in q$, then the equivalent item in t_j is replaced with the value of the corresponding entry in t_i.

- if $\forall s_1, s_2, s_3 \in S' \wedge t \in S''.(s_1, t) \in R(s) \wedge (t, s_2) \in R(p) \wedge (t, s_3) \in R(o)$ and for each triple $t_i = (x_i, y_i, z_i) \in q$ if $s_1 \in L(x_i) \wedge s_2 \in L(y_i)$, then $(x_i, y_i, c_i) \in G$ where c_i is a fresh constant. This case holds if z_i is a non-distinguished variable. Similarly, the case when x_i or y_i or both are variables can be worked out.

- if $\forall s_1, s_2, s_3 \in S' \wedge t \in S''.(s_1, t) \in R(s) \wedge (t, s_2) \in R(p) \wedge (t, s_3) \in R(o)$ and for each triple $t_i = (x_i, e_i, z_i) \in q$ if $s_1 \in [\![\langle s\rangle\langle p\rangle e_i]\!]^K \in L(e_i) \wedge s_2 \in L(e_i) \wedge s_3 \in [\![\langle \bar{o}\rangle\langle p\rangle e_i]\!]^K$, then $(c_i, e_i, d_i) \in G$ where c_i and d_i are fresh constants. This case holds if x_i and y_i are multiply occurring non-distinguished variables. Similarly, all the other cases can be worked out.

Since G is a technical construction obtained from an RDF transition system associated to q, it holds that $[\![q]\!]_G \neq \emptyset$. □

Theorem 1 (Containment under OWL-$\mathcal{ALCH}$ DS entailment)**.** *Given SPARQL queries $q(W)$, $q'(W)$, and a set of $\mathcal{ALCH}$ axioms $\mathcal{S}$, $q(W) \sqsubseteq_{\text{DS}} q'(W)$ iff $\Phi(q \sqsubseteq_{\text{DS}} q')$ is unsatisfiable.*

Proof. ($\Rightarrow$) We show the contrapositive: if $q \not\sqsubseteq_{\text{DS}} q'$, then $\Phi(q \sqsubseteq_{\text{DS}} q')$ is satisfiable. One can verify that every model G of $\mathcal{S}$ in which there is at least one tuple satisfying q but not q' can be turned into an RDF transition system model for $\Phi(q \sqsubseteq_{\text{DS}} q')$. To do so, consider a graph G that satisfies schema axioms $\mathcal{S}$. Assume also that there is a tuple $\overrightarrow{a}$ in the answers of q over G but not in the answers of q'. We can construct an RDF transition system K from G (as done [9]. From Lemma 1, we obtain that $\eta(\mathcal{S})$ is satisfiable in K. At this point, it remains to verify that while $\mathcal{A}(q)$ is satisfiable in K, $\mathcal{A}(q')$ is not. To do so, we build the formulas $\mathcal{A}(q)$ and $\mathcal{A}(q')$ by first skolemizing the distinguished variables using the answer tuple $\overrightarrow{a}$. Consequently, from Lemma 2 one obtains $\mathcal{A}(q)$ is satisfiable. However, $\mathcal{A}(q')$ is unsatisfiable, this is because the nominals in the formula corresponding to the constants and variables that do not appear in the SELECT clause are not satisfied in K. This is justified by the fact that if a formula φ is satisfiable in an RDF transition system, then its negation $\neg\varphi$ is unsatisfiable. So far we have: $\eta(\mathcal{S})$, $\mathcal{A}(q)$, and $\neg\mathcal{A}(q')$ are satisfiable in K. Thus, $\Phi(q \sqsubseteq_{\text{DS}} q')$ is satisfiable in K. Without loss of generality, we get that $\Phi(q \sqsubseteq_{\text{DS}} q')$ is satisfiable.

($\Leftarrow$) $\Phi(q \sqsubseteq_{\text{DS}} q')$ implies that there exists a transition system where the formula $\Phi(q \sqsubseteq_{\text{DS}} q')$ holds. Consequently, K is an RDF transition system due to the restriction imposed on the formula (cf. Proposition 1 [9]). From K it is possible to construct a model $\mathcal{I}$ so that we can utilize Lemma 1 to verify that indeed $\mathcal{I}$ is a model of $\mathcal{S}$. Thus, it remains to show that the answers of q are not included in the answers of q' over $\mathcal{I}$. From our assumption, we have that $\mathcal{A}(q) \wedge \neg\mathcal{A}(q')$ is satisfiable in K. From this, we obtain that $\mathcal{A}(q)$ is satisfiable while $\mathcal{A}(q')$ is not. It is possible to build an RDF graph from the model $\mathcal{I}$ using a function that uses assertions to

form triples. Thus, we have that the answers of q over G are not empty but q' is empty because G contains all those triples that satisfy q and not q'. Therefore, we get that the answers of q are not contained in that of q'. □

Queries and formulas are linearly encoded into the μ-calculus and the satisfiability test of a μ-caclulus formula can be done in an exponential amount of time. Therefore, we get the following proposition.

Proposition 1. *SPARQL query containment under the OWL-$\mathcal{ALCH}$ DS entailment can be solved in a time of $2^{\mathcal{O}(n)}$, where n is the size of the encoding.*

We thus obtain an ExpTime upper bound by Theorem 1. A lower bound is obtained by a reduction from query answering in $\mathcal{ALCH}$ [31]. Hence, proving *optimality* of the complexity bound. So far, we have seen containment under the DS. Next, we address containment under the RBS entailment regime.

Lemma 3 ([4])**.** *Given queries q and q', and an $\mathcal{ALCH}$ TBox $\mathcal{T}$, query containment under OWL Direct Semantics, $q \sqsubseteq_{\text{DS}} q'$, can be polynomially reduced to query entailment with respect to a knowledge base that has a frozen q as an ABox, $\mathcal{K} = \langle \mathcal{T}, f(q)\rangle \models q'$. $f(q)$ is the frozen q i.e., all the terms in the query become constants.*

Theorem 2. *Query containment under $\mathcal{ALCH}$ Direct Semantics is* ExpTime*-hard.*

Proof. To prove this, we reduce the problem of (union of) conjunctive query entailment to query containment with respect to $\mathcal{ALCH}$ axioms [4, 31]. In doing so, we use a function π that translates union of conjunctive queries into SPARQL queries.

$$\begin{aligned}
\pi(q_1 \vee \cdots \vee q_n) &= \pi(q_1) \text{ UNION } \cdots \text{ UNION } \pi(q_n)\\
\pi(C(x)) &= (x, \texttt{type}, C)\\
\pi(R(x,y)) &= (x, R, y)\\
\pi(q_1, \ldots, q_n) &= \pi(q_1) \text{ AND } \cdots \text{ AND } \pi(q_n)
\end{aligned}$$

It remains to show that, given a TBox $\mathcal{T}$ and (union of) conjunctive queries q and q',

$$\pi(q) \sqsubseteq_{\mathcal{T}} \pi(q') \Leftrightarrow \mathcal{K} = \langle \mathcal{T}, f(q)\rangle \models q'$$

($\Rightarrow$) this direction is immediate from [4]. There, it has been implicitly shown that query containment under TBox axioms can be reduced to query entailment w.r.t. a knowledge base (cf. Lemma 3).
($\Leftarrow$) $\langle \mathcal{T}, f(q)\rangle \models q'$

$$\begin{aligned}
&\Rightarrow \forall \mathcal{I}.\big(\mathcal{I} \models \mathcal{T} \text{ and } \mathcal{I} \models f(q) \Rightarrow \mathcal{I} \models q'\big)\\
&\Rightarrow \forall \mathcal{I}.\big(\mathcal{I} \models \mathcal{T} \text{ and } [\![\pi(q)]\!]_G \neq \emptyset \Rightarrow [\![\pi(q')]\!]_G \neq \emptyset\big)^*\\
&\Rightarrow \forall \mathcal{I}.\big(\mathcal{I} \models \mathcal{T} \text{ and } [\![\pi(q)]\!]_G \subseteq [\![\pi(q')]\!]_G\big)\\
&\Rightarrow \pi(q) \sqsubseteq_{\mathcal{T}} \pi(q')
\end{aligned}$$

* G can be constructed from $\mathcal{I}$ using σ', by translating each assertion of the form $a \in A^{\mathcal{I}}$ into $(a, \texttt{type}, A) \in G$ and $(a,b) \in R^{\mathcal{I}}$ into $(a, R, b) \in G$. Furthermore, the knowledge base $f(q)$ can be unfrozen to q. Now, let us verify that, if there is an interpretation for a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{O}\rangle$, then the following holds:

$$\forall \mathcal{I}.\big(\mathcal{I} \models q \Leftrightarrow [\![\pi(q)]\!]_{G=\sigma'(I)} \neq \emptyset\big)$$

this can be proved by induction on the structure of the query.
(Base case) when $q(v) = C(v)$.
($\Rightarrow$) $\mathcal{I} \models C(v) \Rightarrow \tau(v) \in C^{\mathcal{I}}$. Hence, we get $G = \sigma'(I) = \{(\tau(v), \texttt{type}, C)\}$. Clearly, $[\![\pi(C(v))]\!]_G = [\![(v, \texttt{type}, C)]\!]_G \neq \emptyset$.
($\Leftarrow$) Assume $[\![\pi(C(v))]\!]_G \neq \emptyset$

$$\begin{aligned}
&\Rightarrow \exists \rho.\rho \in [\![\pi(C(v))]\!]_G \\
&\Rightarrow \exists \rho.\rho \in [\![(v, \texttt{type}, C)]\!]_G \\
&\Rightarrow \exists \rho.(\rho(v), \rho(\texttt{type}), \rho(C)) \in G \\
&\Rightarrow \exists \rho.(\rho(v), \texttt{type}, C) \in G
\end{aligned}$$

From G, one can generate a DL interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that $C^{\mathcal{I}} = \{\rho(v)\}$. Thus, $\mathcal{I} \models C(v)$ since $\tau(v) = \rho(v) \in C^{\mathcal{I}}$.
When $q(v, v') = R(v, v')$.
($\Rightarrow$) $\mathcal{I} \models R(v, v') \Rightarrow (\tau(v), \tau(v')) \in R^{\mathcal{I}}$. Let us construct a graph G using $\mathcal{I}$, G at least contains $\{(\tau(v), R, \tau(v'))\}$. Obvious, $[\![\pi(R(v, v'))]\!]_G = [\![(v, R, v')]\!]_G \neq \emptyset$.
($\Leftarrow$) Assume $[\![\pi(R(v, v'))]\!]_G$

$$\begin{aligned}
&\Rightarrow \exists \rho.\rho \in [\![(v, R, v')]\!]_G \\
&\Rightarrow \exists \rho.(\rho(v), \rho(R), \rho(v')) \in G \\
&\Rightarrow \exists \rho.(\rho(v), R, \rho(v')) \in G
\end{aligned}$$

Using G, let us build a DL interpretation $\mathcal{I}$ such that $R^{\mathcal{I}} = \{(\rho(v), \rho(v'))\}$. As a consequence, $\mathcal{I} \models^{\tau} R(v, v')$ since $(\tau(v), \tau(v')) = (\rho(v), \rho(v')) \in R^{\mathcal{I}}$. This concludes the proof of the base case.
(Inductive case) when $q(\overrightarrow{v}) = q_1, \ldots, q_n$. Assume that $\mathcal{I} \models q_1, \ldots, q_n$

$$\begin{aligned}
&\Leftrightarrow \mathcal{I} \models^{\tau} q_1 \text{ and } \ldots \text{ and } \mathcal{I} \models^{\tau} q_n \\
&\Leftrightarrow \exists \rho.\rho \in [\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \text{ and } \ldots \text{ and } \rho \in [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})} \text{ by induction hypothesis} \\
&\Leftrightarrow \exists \rho.\rho \in [\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \bowtie \cdots \bowtie [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})} \\
&\Leftrightarrow \exists \rho.\rho \in [\![\pi(q_1) \text{ AND } \cdots \text{ AND } \pi(q_2)]\!]_{\sigma'(\mathcal{I})} \\
&\Leftrightarrow [\![\pi(q_1) \text{ AND } \cdots \text{ AND } \pi(q_2)]\!]_{\sigma'(\mathcal{I})} \neq \emptyset
\end{aligned}$$

When $q(\overrightarrow{v}) = q_1 \vee \cdots \vee q_n$. Starting from $\mathcal{I} \models q_1 \vee \cdots \vee q_n$

$$\begin{aligned}
&\Leftrightarrow \mathcal{I} \models^{\tau} q_1 \text{ or } \ldots \text{ or } \mathcal{I} \models^{\tau} q_n \\
&\Leftrightarrow \exists \rho.\rho \in [\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \text{ or } \ldots \text{ or } \rho \in [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})} \text{from induction hypothesis.} \\
&\Leftrightarrow \exists \rho.\rho \in [\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \cup \cdots \cup \rho \in [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})} \\
&\Leftrightarrow \exists \rho.\rho \in \left([\![\pi(q_1)]\!]_{\sigma'(\mathcal{I})} \cup \cdots \cup [\![\pi(q_n)]\!]_{\sigma'(\mathcal{I})}\right) \\
&\Leftrightarrow \exists \rho.\rho \in \left([\![\pi(q_1) \text{ UNION } \cdots \text{ UNION } \pi(q_n)]\!]_{\sigma'(\mathcal{I})}\right) \\
&\Leftrightarrow [\![\pi(q_1) \text{ UNION } \cdots \text{ UNION } \pi(q_n)]\!]_{\sigma'(\mathcal{I})} \neq \emptyset
\end{aligned}$$

This concludes the induction step and the overall proof. □

3.2 Containment under OWL-$\mathcal{ALCH}$ RDF-Based Semantics Entailment Regime

Unlike DS, in RBS, we need to take care of the RDFS axiomatic triples. Consider the following example.

Example 9. *Consider containment of the following queries.*

```
q  = SELECT ?x WHERE {
        ?x rdf:type rdf:Property .
     }
q' = SELECT ?x WHERE {
        ?x rdf:type
           rdfs:ContainerMembershipProperty .
     }
```

The answers of these queries under the OWL-$\mathcal{ALCH}$ RBS are infinite due to the RDFS axiomatic triples (particularly rdf:_i). As answering is tied to a queried graph, this can be eliminated as discussed above. However, containment faces a challenge as it is impossible to encode infinite axiomatic triples. One approach to alleviate this problem is to avoid all axiomatic triples that involve the URI rdf:_i. This is what is done is this paper. Another approach is by analysing the queries beforehand to see if they contain vocabularies that lead to infinite answers. When so, restrict the axiomatic triples to those that occur in the queries. As a consequence, $q \not\sqsubseteq_{\text{RBS}} q'$ and $q' \sqsubseteq_{\text{RBS}} q$. By contrast, $q \not\sqsubseteq_{\text{DS}} q'$ and $q' \not\sqsubseteq_{\text{DS}} q$ since the DS does not involve axiomatic triples.

SPARQL query containment under the RBS entailment regime can be reduced to unsatisfiability test in the μ-calculus. That is, by encoding queries, OWL-$\mathcal{ALCH}$ schema axioms, and RDFS axiomatic triples into the μ-calculus, and testing the validity of the reduction. It is possible to reuse the encodings for the DS entailment regime with the encoding of axiomatic triples. Instead, here, we provide a different encoding for the semantics of the schema using OWL inference rules and the schema itself.

Definition 7. *The encoding of RDFS axiomatic triples (AT) minus rdf:_i for $i \in \mathbb{N}$, $AT = \{at_1, \cdots, at_n\}$, is produced by encoding each triple $at_i = (x, y, z) \in AT$ such that:*

$$\Phi_{\text{AT}} = \bigwedge_{i=1 \wedge at_i \in AT}^{n} lfp\big(X, \langle \bar{s} \rangle x \wedge \langle p \rangle y \wedge \langle o \rangle z\big)$$

where x, y, and z are atomic propositions encoding triple elements.

3.2.1 Encoding OWL-$\mathcal{ALCH}$ rules and schema

The W3C calculus provides 78 rules, various simplification rules are introduced that allow to restrict to a much smaller number of features [23]. Out of which, we are mainly interested in the rules that express the semantics of the schema vocabulary[4]. So that this semantics can be taken into account when testing containment. Note that, transitive and reflexive properties such as `sc`, `sp`, `eqp`, `eqc`, and `sameAs` are encoded as reflexive transitive closure: `sc`* and so on. This is because, with the μ-calculus, it is not possible to enforce transitivity of a property in a transition system. This is a consequence of the tree-model property of the μ-calculus, and the fact that transitivity does not hold in tree-shaped models [Sattler & Vardi 2001].

Definition 8. *The encoding of OWL rules into μ-calculus formulae, Φ_{rules}, is given by a function γ as shown in Table 4.*

$$\Phi_{\text{rules}} = \bigvee_{i=1 \wedge t_i \in Rules}^{n} \gamma(rule_i)$$

[4]Table 9 in `http://www.w3.org/TR/owl2-profiles/`

name	Rule	μ-calculus encoding
rdfp5a	$(a,b,c) \Rightarrow (a,\texttt{sameAs},a)$	$gfp\big(X, \theta(a,b,c) \Rightarrow \theta(a,\texttt{sameAs},a)\big)$
rdfp5b	$(a,b,c) \Rightarrow (a,\texttt{sameAs},a)$	$gfp\big(X, \theta(a,b,c) \Rightarrow \theta(c,\texttt{sameAs},c)\big)$
rdfp6	$(a,\texttt{sameAs},c) \Rightarrow (c,\texttt{sameAs},a)$	$gfp\big(X, \theta(a,\texttt{sameAs},c) \Rightarrow \theta(c,\texttt{sameAs},a)\big)$
rdfp7	$(a,\texttt{sameAs},b).(b,\texttt{sameAs},c) \Rightarrow$ $(a,\texttt{sameAs},c)$	$gfp\big(X, \theta(a,\texttt{sameAs},c) \Rightarrow \theta(a,\texttt{sameAs}^+,c)\big)*$
rdfp9	$(a,\texttt{type},\texttt{Class}).(a,\texttt{sameAs},c) \Rightarrow$ $(a,\texttt{sc},c)$	$gfp\big(X, \theta(a,\texttt{type},\texttt{Class}) \wedge \theta(a,\texttt{sameAs},c)$ $\Rightarrow \theta(a,\texttt{sc},c)\big)$
rdfp10	$(b,\texttt{type},\texttt{Property}).$ $(b,\texttt{sameAs},d) \Rightarrow (b,\texttt{sp},d)$	$gfp\big(X, \theta(b,\texttt{type},\texttt{Property}) \wedge$ $\theta(b,\texttt{sameAs},d) \Rightarrow \theta(b,\texttt{sp},d)\big)$
rdfp12a	$(a,\texttt{eqc},c) \Rightarrow (a,\texttt{sc},c)$	$gfp\big(X, \theta(a,\texttt{eqc},c) \Rightarrow \theta(a,\texttt{sc},c)\big)$
rdfp12b	$(a,\texttt{eqc},c) \Rightarrow (c,\texttt{sc},a)$	$gfp\big(X, \theta(a,\texttt{eqc},c) \Rightarrow \theta(c,\texttt{sc},a)\big)$
rdfp12c	$(a,\texttt{sc},b).(b,\texttt{sc},c) \Rightarrow (a,\texttt{eqc},c)$	$gfp\big(X, \theta(a,\texttt{sc},b) \wedge \theta(b,\texttt{sc},c) \Rightarrow \theta(a,\texttt{eqc}^*,c)\big)*$
rdfp13a	$(a,\texttt{eqp},c) \Rightarrow (a,\texttt{sp},c)$	$gfp\big(X, \theta(a,\texttt{eqp},c) \Rightarrow \theta(a,\texttt{sp},c)\big)$
rdfp13b	$(a,\texttt{eqp},c) \Rightarrow (c,\texttt{sp},a)$	$gfp\big(X, \theta(a,\texttt{eqp},c) \Rightarrow \theta(c,\texttt{sp},a)\big)$
rdfp13c	$(a,\texttt{sp},c).(c,\texttt{sp},a) \Rightarrow (a,\texttt{eqp},c)$	$gfp\big(X, \theta(a,\texttt{sp},c) \wedge \theta(c,\texttt{sp},a) \Rightarrow \theta(a,\texttt{eqp},c)\big)*$
rdfp15	$(a,\texttt{svf},b).(a,\texttt{onp},c).$ $(d,c,e).(e,\texttt{type},b) \Rightarrow (d,\texttt{type},a)$	$\theta(d,\theta'(\theta(a,\texttt{svf},\theta'(e,\texttt{type},b)),\texttt{onp},c),b) \Rightarrow$ $\theta(d,\texttt{type},a)$
rdfp16	$(a,\texttt{avf},b).(a,\texttt{onp},c).$ $(d,\texttt{type},a).(d,c,e) \Rightarrow (e,\texttt{type},b)$	$\theta'(\theta(d,\texttt{type},\theta(a,\texttt{onp},c) \wedge$ $\theta(a,\texttt{avf},b)),c,e) \Rightarrow (e,\texttt{type},b)$
prp-eqp1	$(a,\texttt{eqp},b)(c,a,d) \Rightarrow (c,b,d)$	$gfp\big(X, \theta(c,\theta(a,\texttt{eqp},b),d) \Rightarrow \theta(c,b,d)\big)$
prp-eqp2	$(a,\texttt{eqp},b)(c,b,d) \Rightarrow (c,a,d)$	$gfp\big(X, \theta(c,\theta'(a,\texttt{eqp},b),d) \Rightarrow \theta(c,b,d)\big)$
prp-pdw	$(a,\texttt{propDisWith},b).(c,a,d)$. $(c,b,d) \Rightarrow \texttt{false}$	$gfp\big(X, \theta(c,\theta(a,\texttt{propDisWith},b),d)$ $\wedge \theta(c,\theta'(a,\texttt{propDisWith},b),d) \Rightarrow \bot\big)$
cls-nothing2	$(a,\texttt{type},\texttt{Nothing}) \Rightarrow \texttt{false}$	$gfp\big(X, \theta(a,\texttt{type},\texttt{Nothing}) \Rightarrow \bot\big)$
cls-svf1	$(a,\texttt{svf},b)$ $(a,\texttt{onp},c)$ $(d,\texttt{type},a)$ $(d,c,e)(e,\texttt{type},b) \Rightarrow (d,\texttt{type},a)$	$gfp\big(X, \theta(d,\theta'(\theta(a,\texttt{svf},\theta'(e,\texttt{type},b)),\texttt{onp},c),b) \Rightarrow$ $\theta(d,\texttt{type},a)\big)$
cls-avf	$(a,\texttt{avf},b)$ $(a,\texttt{onp},c)$ $(d,\texttt{type},a)$ $(d,c,e) \Rightarrow (e,\texttt{type},b)$	$gfp\big(X, \theta'(\theta(d,\texttt{type},\theta(a,\texttt{onp},c) \wedge$ $\theta(a,\texttt{avf},b)),c,e) \Rightarrow (e,\texttt{type},b)\big)$
scm-sco	$(a,\texttt{sc},b)(b,\texttt{sc},c) \Rightarrow (a,\texttt{sc},c)$	$gfp\big(X, \mu Y.\langle p\rangle\texttt{sc} \wedge \langle o\rangle\top \vee \langle p\rangle\texttt{sc} \wedge \langle s\rangle Y\big)$
scm-spo	$(a,\texttt{sp},b)(b,\texttt{sp},c) \Rightarrow (a,\texttt{sp},c)$	$gfp\big(X, \mu Y.\langle p\rangle\texttt{sp} \wedge \langle o\rangle\top \vee \langle p\rangle\texttt{sp} \wedge \langle s\rangle Y\big)$
scm-dom1	$(a,\texttt{dom},b)(b,\texttt{sc},d) \Rightarrow (a,\texttt{dom},d)$	$gfp\big(X, \theta(a,\texttt{dom},\theta(b,\texttt{sc},d)) \Rightarrow \theta(a,\texttt{dom},d)\big)$
scm-dom2	$(a,\texttt{dom},b)(d,\texttt{sp},a) \Rightarrow (d,\texttt{dom},b)$	$gfp\big(X, \theta(d,\texttt{sp},\theta(a,\texttt{dom},b)) \Rightarrow \theta(d,\texttt{dom},b)\big)$
scm-rng1	$(a,\texttt{range},b)(b,\texttt{sc},d) \Rightarrow$ $(a,\texttt{range},d)$	$gfp\big(X, \theta(a,\texttt{range},\theta(b,\texttt{sc},d)) \Rightarrow \theta(a,\texttt{range},d)\big)$
scm-rng2	$(a,\texttt{range},b)(d,\texttt{sp},a) \Rightarrow$ $(d,\texttt{range},b)$	$gfp\big(X, \theta(d,\texttt{sp},\theta(a,\texttt{range},b)) \Rightarrow \theta(d,\texttt{range},b)\big)$
		where $\theta((x,y,z)) = x \wedge \langle s\rangle(\langle p\rangle y \wedge \langle o\rangle z)$ and $\theta'((x,y,z)) = z \wedge \langle \bar{o}\rangle(\langle p\rangle y \wedge \langle \bar{s}\rangle x$

Table 4: OWL 2 RL^{min} rules encoded into μ-calculus formulae.

Definition 9. *The encoding of an OWL-$\mathcal{ALCH}$ TBox schema $S = \{t_1, \cdots, t_n\}$ is produced by encoding each schema triple $t_i = (x,y,z) \in S$ such that:*

$$\Phi_S = \bigwedge_{i=1 \wedge t_i \in S}^{n} \big(lfp\big(X, \langle \bar{s} \rangle x \wedge \langle p \rangle y \wedge \langle o \rangle z\big)\big)$$

where x, y, and z are atomic propositions corresponding to triple elements.

So far, we have produced the encoding of the schema, inference rules, axiomatic triples and queries. Thus, we are ready to prove the correctness of the reduction:

$$\Phi(q \sqsubseteq_{\text{RBS}} q') = \Phi_{\text{AT}} \wedge \Phi_{\text{rules}} \wedge \Phi_{\mathcal{S}} \wedge \mathcal{A}(q) \wedge \neg \mathcal{A}(q')$$

Theorem 3 (Containment under OWL-$\mathcal{ALCH}$ RBS)**.** *Given SPARQL queries $q(W)$, $q'(W)$, and a set of $\mathcal{ALCH}$ axioms $\mathcal{S}$, $q(W) \sqsubseteq_{\text{RBS}} q'(W)$ iff $\Phi(q \sqsubseteq_{\text{RBS}} q')$ is unsatisfiable.*

Proof. It is sufficient to extend the proof of theorem 1. Since the RDFS axiomatic triples are satisfied by every graph, one can build an RDF transition system from such a graph which can be a model for Φ_{AT}. Thus, we get that; when the axiomatic triples are satisfiable, then Φ_{AT} is also satisfiable. □

SPARQL query containment under RBS entailment can be determined in exponential amount of time (upper bound). Thus far we have shown that SPARQL query containment under the DS and RBS entailment regimes can be determined with the μ-calculus. Unlike DS, RBS adds RDFS axiomatic triples (making the size of the encoding large) and allows higher order queries [14].

4 Discussion

Containment under simple, RDF, and RDFS entailment In order to determine SPARQL containment under simple, RDF, and RDFS entailment regimes. We fall back on the definition of entailment based RDF(S) semantics [17, 34]. To be able to use this semantics for testing containment, we need to transform UCQs into RDF graphs by replacing variables with blank nodes. From any query it is possible to build an homomorphic graph by collecting all triples connected by AND and only those at the left of UNION (replacing variables by blank nodes). This graph is consistent as all RDF graphs. This is only true if the query does not contain a variable in the predicate position of its triple patterns. However, it has been shown that allowing blank nodes in predicate positions makes RDFS reasoning complete [34]. Leading us to the notion of *generalized RDF graph*, it is defined as a subset of the set UB × UB × UBL.

Definition 10 (BGP containment)**.** *Given two BGPs q and q', determining $q \sqsubseteq_s q'$, $q \sqsubseteq_{\text{rdf}} q'$, and $q \sqsubseteq_{\text{rdfs}} q'$ can be done by verifying the entailments $f(q) \models_s g(q')$, $f(q) \models_{\text{rdf}} g(q')$, and $f(q) \models_{\text{rdfs}} g(q')$ respectively. f and g transform each triple pattern in q and q' as:*

$$f((s,p,o)) = (f(s), f(p), f(o)) \qquad g((s,p,o)) = (g(s), g(p), g(o))$$

$$f(a) = \begin{cases} freshURI(a) \text{ if } a \in var(q) \\ a \text{ otherwise} \end{cases} \qquad g(a) = \begin{cases} \_:a \text{ if } a \in var(q) \\ a \text{ otherwise} \end{cases}$$

Definition 11 (Entailment)**.** *Given query graphs G and G', G e-entails G', written as $G \models_e G'$, if G can be extended to a graph G_{ex} by applying e-entailment rules such that $G' \subseteq G_{\text{ex}}$.*

To define the containment between two UNION patterns, we extend the functions f and g as: $g(q_1 \text{ UNION } q_2) = g(q_1), g(q_2)$.

Definition 12 (UNION pattern containment)**.** *Given two UNION patterns q and q', let $f(q) = \{g_1, \ldots, g_n\}$ and $g(q') = \{g'_1, \ldots, g'_m\}$ be their respective transformations, $q \sqsubseteq_e q'$ iff $\forall g'_i \in g(q')\ \exists g_j \in f(q) : g'_i \models_e g_j$, where $1 \leq i, j \leq n$.*

Since the transformation of a BGP into an RDF graph is linear in the size of the pattern, the complexity of deciding containment under entailment regimes is the same as entailment check.

5 Related Work

Recently, the study of SPARQL query containment has gained moment, notably in [7, 8, 10, 25]. The literatures in [7] and [8] address containment under the ρdf [29] entailment regime and $\mathcal{SHI}$ schema axioms respectively and establish a double exponential upper bound complexity. Additionally, in [25] the containment and optimization of OPTIONAL queries is investigated while providing a Π_2^P-complete complexity for containment. Importantly, a benchmark of containment solvers is revealed in [10]. The implementations of containment solvers allow the extension of query and ontology languages. Thus, what we did here can benefit from these implementations.

Query Entailment is the decision problem associated with query answering. For CQs, query answering and containment are equivalent problems. In fact, query containment can be reduced to query answering. In this regard, conjunctive query containment under the description logic $\mathcal{DLR}$ is studied in [4]. CQ query answering in the presence of simple ontologies (fragments of DL-Lite) has been studied [3, 27]. For expressive ontology languages, query entailment (and hence containment) in DLs ranging from $\mathcal{ALCI}$ to $\mathcal{SHIQ}$ is shown to be 2ExpTime in [11, 15, 26, 30]. By contrast, in this study we do not deal with the same query language as the one dealt with in [15, 31]. In fact, the supported SPARQL fragment is strictly larger than the one studied in [15, 31]. Specifically, UCQs in [31] are made of $C(x), R(x, y)$ for an atom C, a role R, and variables x and y, whereas we do also support queries capable of querying concept and role names at the same time, such as $q(x) = (x, y, z)$. Beyond this, the novelty of the study is the reduction of the SPARQL containment problem to μ-calculus satisfiability, and the advantages of using such a logic: expressivity, good computational properties, extensibility. The main focus of the contribution is not the complexity bound by itself but rather a new approach with a broader logic, paving the way for future extensions as it was never done before.

Finally, with an implicit goal of minimizing query evaluation costs, in [32] comprehensive complexity results were obtained for the problem of redundancy elimination on RDF graphs in the presence of rules (RDFS or OWL), constraints (tuple-generating dependencies) and with respect to SPARQL queries.

6 Conclusion

We showed that deciding the containment of SPARQL queries under the OWL-$\mathcal{ALCH}$ DS and RBS semantics is ExpTime-complete. We have reduced the problem to the validity test in the μ-calculus to establish membership and then the hardness by a reduction from query answering in $\mathcal{ALCH}$. Implicitly, we have also shown that containment of SPARQL path queries under $\mathcal{ALCH}$ axioms can be determined in exponential amount of time. Recently, some implementations of containment solvers have been made available [10, 25]. Some of these are built on top of μ-calculus satisfiability solvers, thus this study enables to take advantage of these implementations. In the future, we will experiment the proposed approach using these tools. Additionally, we will explore the possibility of extending the schema language by using other fragments of the μ-calculus that enable encoding number restrictions [2].

A summary of complexity results on the problem of SPARQL query containment is shown in Table 5.

	$\sqsubseteq_{\text{rdf}}$, $\sqsubseteq_{\text{rdfs}}$	$\sqsubseteq_s$	$\sqsubseteq_{\text{DS}}$	$\sqsubseteq_{\text{RBS}}$
BGP, UNION	NP-complete	NP-complete	new (ExpTime)	new (ExpTime)
CQ, UCQ		NP-complete	2ExpTime	2ExpTime
OPTIONAL		Π_2^P-complete		
OPTIONAL with projection		Π_2^P		

Table 5: SPARQL query containment state-of-the-art.

References

[1] F. Alkhateeb, J.F. Baget, and J. Euzenat. Extending SPARQL with regular expression patterns (for querying RDF). *J. Web Semantics*, 7(2):57–73, 2009.

[2] P. A. Bonatti, C. Lutz, A. Murano, and M. Y. Vardi. The Complexity of Enriched μ-calculi. *Automata, Languages and Programming*, pages 540–551, 2006.

[3] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. Conjunctive Query Containment and Answering under Description Logics Constraints. *ACM Trans. on Computational Logic*, 9(3):22.1–22.31, 2008.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of Conjunctive Regular Path Queries with Inverse. In *Proc. of the 7th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 176–185, 2000.

[6] A. K. Chandra and P. M. Merlin. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 77–90. ACM, 1977.

[7] M. W. Chekol, J. Euzenat, P. Genevès, and N. Layaïda. SPARQL query containment under RDFS entailment regime. In *IJCAR'12*, pages 134–148. Springer, 2012.

[8] M. W. Chekol, J. Euzenat, P. Genevès, and N. Layaïda. SPARQL query containment under SHI axioms. In *AAAI'12*, volume 1, pages 10–16, 2012.

[9] Melisachew Wudage Chekol. *Static Analysis of Semantic Web Queries*. PhD thesis, Université de Grenoble, 2012.

[10] Melisachew Wudage Chekol, Jérôme Euzenat, Pierre Genevès, and Nabil Layaïda. Evaluating and benchmarking sparql query containment solvers. In *International Semantic Web Conference (2)*, pages 408–423, 2013.

[11] T. Eiter, C. Lutz, M. Ortiz, and M. Šimkus. Query answering in description logics with transitive roles. In *Proc. of IJCAI*, pages 759–764, 2009.

[12] P. Genevès, N. Layaïda, and A. Schmitt. Efficient Static Analysis of XML Paths and Types. In *PLDI '07*, pages 342–351, New York, NY, USA, 2007. ACM.

[13] Pierre Genevès and Nabil Layaïda. A system for the static analysis of XPath. *ACM Trans. Inf. Syst.*, 24(4):475–502, 2006.

[14] B. Glimm. Using SPARQL with RDFS and OWL entailment. *Reasoning Web. Semantic Technologies for the Web of Data*, pages 137–201, 2011.

[15] B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic *shiq*. *J Artif Intell Res*, 31:157–204, 2008.

[16] B. Glimm and M. Krötzsch. SPARQL beyond subgraph matching. *The Semantic Web–ISWC 2010*, pages 241–256, 2010.

[17] P. Hayes. RDF semantics. W3C Recommendation, 2004.

[18] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, and Sebastian Rudolph. Owl 2 web ontology language primer. *W3C recommendation*, 27:1–123, 2009.

[19] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Logic for Programming and Automated Reasoning*, pages 161–180. Springer, 1999.

[20] Y.E. Ioannidis and R. Ramakrishnan. Containment of conjunctive queries: Beyond relations as sets. *ACM Transactions on Database Systems (TODS)*, 20(3):288–324, 1995.

[21] I. Kollia, B. Glimm, and I. Horrocks. SPARQL query answering over OWL ontologies. In *Proc. 8th ESWC, Heraklion (GR)*, volume 6643 of *LNCS*, pages 382–396, 2011.

[22] D. Kozen. Results on the propositional μ-calculus. *Theor. Comp. Sci.*, 27:333–354, 1983.

[23] M. Krötzsch. The not-so-easy task of computing class subsumptions in OWL RL. In *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*, volume 7649 of *Lecture Notes in Computer Science*, pages 279–294. Springer, 2012.

[24] Er Kubias, Simon Schenk, Steffen Staab, and Jeff Z Pan. Owl saiql-an owl dl query language for ontology extraction. In *In Proc. of OWLED-07*. Citeseer, 2007.

[25] Andrés Letelier, Jorge Pérez, Reinhard Pichler, and Sebastian Skritek. Static analysis and optimization of semantic web queries. *ACM Trans. Database Syst.*, 38(4):25, 2013.

[26] C. Lutz. The complexity of conjunctive query answering in expressive description logics. *Automated Reasoning*, pages 179–193, 2008.

[27] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic el using a relational database system. In *IJCAI*, pages 2070–2075, 2009.

[28] Frank Manola, Eric Miller, and Brian McBride. Rdf primer. *W3C recommendation*, 10:1–107, 2004.

[29] Sergio Muñoz, Jorge Pérez, and Claudio Gutierrez. Minimal deductive systems for RDF. In *The Semantic Web: Research and Applications*, pages 53–67. Springer, 2007.

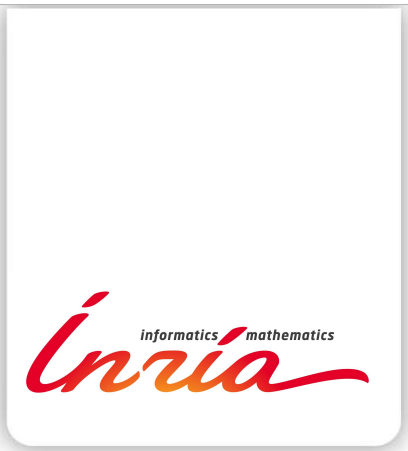
[30] M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning*, 41(1):61–98, 2008.

[31] M. Ortiz, M. Šimkus, and T. Eiter. Worst-case optimal conjunctive query answering for an expressive description logic without inverses. In *Proc. of AAAI*, volume 8, 2008.

[32] R. Pichler, A. Polleres, S. Skritek, and S. Woltran. Redundancy elimination on rdf graphs in the presence of rules, constraints, and queries. *Web Reasoning and Rule Systems*, pages 133–148, 2010.

[33] E. Sirin and B. Parsia. Sparql-dl: Sparql query for owl-dl. In *3rd OWL Experiences and Directions Workshop (OWLED-2007)*, volume 4, 2007.

[34] Herman J ter Horst. Combining rdf and part of owl with rules: Semantics, decidability, complexity. In *The Semantic Web–ISWC 2005*, pages 668–684. Springer, 2005.



RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399